Software Design Document for the Land Information System

Submitted under Task Agreement GSFC-CT-2

Cooperative Agreement Notice (CAN) CAN-00OES-01

Increasing Interoperability and Performance of Grand Challenge Applications in the Earth, Space, Life, and Microgravity Sciences

Version 1.0

Revision history:

Version	***Summary of Changes***	***Date***
1.0	Initial release.	8/13/02

Table of Contents

List of Figures

List of Tables

Acronyms and Terms

ALMA: Assistance for Land-surface Modeling Activities

API: Application Programming Interface

CGI: Common Gateway Interface

CLM: Community Land Model

DODS: Distributed Ocean Data System

ESMF: Earth System Modeling Framework

GrADS: Grid Analysis and Display System

LDAS: Land Data Assimilation System

LIS: Land Information System

MRTG: Multi Router Traffic Grapher

NFS: Network File System

NOAH: National Centers for Environmental Prediction, Oregon State University, United States Air Force, and Office of Hydrology Land Surface Model

PXE: Preboot Execution Environment

RAID: Redundant Array of Inexpensive Disks

SNMP: Simple Network Management Protocol

VIC: Variable Infiltration Capacity Land Surface Model

1 Introduction

This Software Design Document establishes the software design for the Land Information System (LIS). LIS is a project to build a high-resolution, high-performance land surface modeling and data assimilation system to support a wide range of land surface research activities and applications.

This document has been prepared in accordance with the requirements of the Task Agreement GSFC-CT-2 under Cooperative Agreement Notice CAN-00-OES-01 Increasing Interoperability and Performance of Grand Challenge Applications in the Earth, Space, Life, and Microgravity Sciences, funded by NASA's ESTO Computational Technologies (formerly High Performance Computing and Communications) Project.

1.1 Purpose and goals

This document serves as the blueprint for the software development and implementation of the Land Information System (LIS).

The design goals of LIS are near real-time, high-resolution (up to 1km) global land data simulation executed on highly parallel computing platforms, with well defined, standard-conforming interfaces and data structures to interface and inter-operate with other Earth system models, and with flexible and friendly web-based user interfaces.

1.2 Scope

This document covers the design of all the LIS software components for the three-year duration of the LIS project. The document focuses primarily on the implementation of the LIS software on a general-purpose Linux cluster system, and most of the component designs also apply to an SGI Origin 3000 system. This document does not cover design for other hardware/software platforms.

Specifically, this design covers the following aspects of LIS:

- Realistic land surface modeling. LIS will simulate the global land surface variables using various land surface models, driven by atmospheric "forcing data" (e.g., precipitation, radiation, wind speed, temperature, humidity) from various sources.
- High performance computing. LIS will perform high-performance, parallel computing for near real-time, high-resolution land surface modeling research and operations.
- Efficient data management. The high-resolution land surface simulation will produce a huge data throughput, and LIS will retrieve, store, interpolate, re-project, sub-set, and backup the input and output data efficiently.

- Usability. LIS will provide intuitive web-based interfaces to users with varying levels of access to LIS data and system resources, and enforce user security policies.
- Interoperable and portable computing. LIS will incorporate the ALMA (Assistance for Land surface Modeling Activities) and ESMF (Earth System Modeling Framework) standards to facilitate inter-operation with other Earth system models. In order to demonstrate portability of LIS, the land surface modeling component will be implemented on a custom-designed Linux cluster and an SGI Origin 3000.

2 Land Surface Modeling and Data Assimilation

In general, land surface modeling seeks to predict the terrestrial water, energy and biogeochemical processes by solving the governing equations of the soil-vegetation-snowpack medium. Land surface data assimilation seeks to synthesize data and land surface models to improve our ability to predict and understand these processes. The ability to predict terrestrial water, energy and biogeochemical processes is critical for applications in weather and climate prediction, agricultural forecasting, water resources management, hazard mitigation and mobility assessment.

In order to predict water, energy and biogeochemical processes using (typically 1-D vertical) partial differential equations, land surface models require three types of inputs: 1) initial conditions, which describe the initial state of land surface; 2) boundary conditions, which describe both the upper (atmospheric) fluxes or states also known as "forcings" and the lower (soil) fluxes or states; and 3) parameters, which are a function of soil, vegetation, topography, etc., and are used to solve the governing equations.

The proposed LIS framework will include various components that facilitate global land surface modeling within a data assimilation system framework. The main software components of the system are:

- LDAS (Land Data Assimilation System) : is a software system that integrates the use of land surface models in a data assimilation framework.
- Land surface Models : LIS will include 3 different land surface models, namely, CLM, NOAH, and VIC.

These components are explained in detail in the following sections.

2.1 Land Data Assimilation System (LDAS)

LDAS is a model control and input/output system (consisting of a number of subroutines, modules written in Fortran 90 source code) that drives multiple offline one dimensional land surface models (LSMs) using a vegetation defined "tile" or "patch" approach to simulate sub-grid scale variability. The one-dimensional LSMs such as CLM and NOAH, which are subroutines of LDAS, apply the governing equations of the physical processes of the soil-vegetation-snowpack medium. These land surface models aim to characterize the transfer of mass, energy, and momentum between a vegetated surface and the atmosphere.

LDAS makes use of various satellite and ground based observation systems within a land data assimilation framework to produce optimal output fields of land surface states and fluxes. The LSM predictions are greatly improved through the use of a data assimilation environment such as the one provided by LDAS. In addition to being forced with real time output from numerical prediction models and satellite and radar precipitation measurements, LDAS derives model parameters from existing topography, vegetation and soil coverages. The model results are aggregated to various temporal and spatial scales, e.g., 3 hourly, 0.25 deg x 0.25 deg.

The execution of LDAS starts with reading in the user specifications. The user selects the model domain and spatial resolution, the duration and timestep of the run, the land surface model, the type of forcing from a list of model and observation based data sources, the number of ``tiles' per grid square (described below), the soil parameterization scheme, reading and writing of restart files, output specifications, and the functioning of several other enhancements including elevation correction and data assimilation.

The system then reads the vegetation information and assigns subgrid tiles on which to run the one-dimensional simulations. LDAS runs its 1-D land models on vegetation-based "tiles" to simulate variability below the scale of the model grid squares. A tile is not tied to a specific location within the grid square. Each tile represents the area covered by a given vegetation type.

Memory is dynamically allocated to the global variables, many of which exist within Fortran 90 modules. The model parameters are read and computed next. The time loop begins and forcing data is read, time/space interpolation is computed and modified as necessary. Forcing data is used to specify boundary conditions to the land surface model. The LSMs in LDAS are driven by atmospheric forcing data such as precipitation, radiation, wind speed, temperature, humidity, etc., from various sources. LDAS applies spatial interpolation to convert forcing data to the appropriate resolution required by the model. Since the forcing data is read in at certain regular intervals, LDAS also temporally interpolates time average or instantaneous data to that needed by the model at the current timestep. The selected model is run for a vector of ``tiles", intermediate information is stored in modular arrays, and output and restart files are written at the specified output interval.

2.2 Community Land Model (CLM)

CLM (Community Land Model) is a 1-D land surface model, written in Fortran 90, developed by a grass-roots collaboration of scientists who have an interest in making a general land model available for public use. LDAS currently uses CLM version 1.0, formerly known as the Common Land Model. CLM version 2.0 was released in May 2002 and will be implemented in future version of LDAS/LIS. The source code for CLM 2.0 is freely available from the National Center for Atmospheric Research (NCAR) (http://www.cgd.ucar.edu/tss/clm/). The CLM is used as the land model for the Community Climate System Model (CCSM) (http://www.ccsm.ucar.edu/), which includes the Community Atmosphere Model (CAM) (http://www.cgd.ucar.edu/cms/). CLM is executed with all forcing, parameters, dimensioning, output routines, and coupling performed by an external driver of the user's design (in this case done by LDAS). CLM requires pre-processed data such as the land surface type, soil and vegetation parameters, model initialization, and atmospheric boundary conditions as input. The model applies finite-difference spatial discretization methods and a fully implicit time-integration scheme to numerically integrate the governing equations. The model subroutines apply the governing equations of the physical processes of the soil-vegetation-snowpack medium, including the surface energy balance equation, Richards' (1931) equation for soil hydraulics, the diffusion equation for soil heat transfer, the energy-mass balance equation for the snowpack, and the Collatz et al. (1991) formulation for the conductance of canopy transpiration.

2.3 The Community NOAH Land Surface Model

The community NOAH Land Surface Model is a stand-alone, uncoupled, 1-D column model freely available at the National Centers for Environmental Prediction (NCEP; ftp://ftp.ncep.noaa.gov/pub/gcp/ldas/noahlsm/). The name is an acronym representing the various developers of the model (N: NCEP; O: Oregon State University, Dept. of Atmospheric Sciences; A: Air Force (both AFWA and AFRL - formerly AFGL, PL); and H: Hydrologic Research Lab - NWS (now Office of Hydrologic Dev -- OHD)). NOAH can be executed in either coupled or uncoupled mode. It has been coupled with the operational NCEP mesoscale Eta model (Chen et al., 1997) and its companion Eta Data Assimilation System (EDAS) (Rogers et al., 1996), and the NCEP global Medium-Range Forecast model (MRF) and its companion Global Data Assimilation System (GDAS). When NOAH is executed in uncoupled mode, near-surface atmospheric forcing data (e.g., precipitation, radiation, wind speed, temperature, humidity) is required as input. NOAH simulates soil moisture (both liquid and frozen), soil temperature, skin temperature, snowpack depth, snowpack water equivalent, canopy water content, and the energy flux and water flux terms of the surface energy balance and surface water balance. The model applies finite-difference spatial discretization methods and a Crank-Nicholson time-integration scheme to numerically integrate the governing equations of the physical processes of the soil vegetation-snowpack medium, including the surface energy balance equation, Richards’ (1931) equation for soil hydraulics, the diffusion equation for soil heat transfer, the energy-mass balance equation for the snowpack, and the Jarvis (1976) equation for the conductance of canopy transpiration.

2.4 Variable Infiltration Capacity (VIC) Model

Variable Infiltration Capacity (VIC) model is a macroscale hydrologic model, written in C, being developed at the University of Washington and Princeton University. The VIC code repository along with the model description and source code documentation is publicly available at http://www.hydro.washington.edu/Lettenmaier/Models/VIC/VIChome.html. VIC is used in macroscopic land use models such as SEA - BASINS (http://boto.ocean.washington.edu/seasia/intro.htm). VIC is a semi-distributed, grid-based hydrological model, which parameterizes the dominant hydrometeorological processes taking place at the land surface - atmospheric interface. The execution of VIC model requires preprocessed data such as precipitation, temperature, meteorological forcing, soil and vegetation parameters, etc. as input. The model uses three soil layers and one vegetation layer with energy and moisture fluxes exchanged between the layers. The VIC model represents surface and subsurface hydrologic processes on a spatially distributed (grid cell) basis. Partitioning grid cell areas to different vegetation classes can approximate sub-grid scale variation in vegetation characteristics. VIC models the processes governing the flux and storage of water and heat in each cell-sized system of vegetation and soil structure. The water balance portion of VIC is based on three concepts:

1) Division of grid-cell into fraction sub-grid vegetation coverage.
2) The variable infiltration curve for rainfall/runoff partitioning at the land surface.
3) A baseflow/deep soil moisture curve for lateral baseflow.

Water balance calculations are preformed at three soil layers and within a vegetation canopy. An energy balance is calculated at the land surface. A full description of algorithms in VIC can be found in the references listed at the VIC website.

3 LIS software architecture

This section describes the software architecture of the components of LIS. The proposed LIS framework will have the following functional components: (1) A system for high resolution global land data assimilation system, involving several land surface models, and land data assimilation technologies. (2) A web-based user interface that accesses data mining, numerical modeling and visualization tools. To facilitate these features, LIS will integrate the use of various software systems such as LDAS, land surface models, GrADS – DODS, etc. LIS is also expected to act as a framework that enables the land surface modeling community to define new standards and also to assist in the definition and demonstration of the ESMF. As a result, the design of LIS will also feature the incorporation of new standards and specifications such as ALMA and ESMF.

Figure 1 shows the LDAS software architecture. As mentioned earlier, currently LDAS includes CLM and NOAH land surface models. VIC land surface model will be incorporated in the future versions of LDAS and LIS.

Figure 2 presents the LIS software architecture. It can be noticed that LIS will be built upon the existing LDAS, with new components and expanded functionalities for the support of parallel processing, GrADS-DODS server-based data management, ALMA and ESMF-compliance, web-based user interfaces, and system management of a Linux cluster platform

The function of LIS dictates a highly modular system design and requires all the modules, or components, to work together smoothly and reliably. Figure 2 shows an overview of the LIS software architecture and its components, and their interactions. LIS will continuously take in relevant atmospheric observational data, and will subsequently use it to force the land surface models, and the land surface simulation is carried out in a highly parallel fashion. Meanwhile the large amount of output data will be efficiently managed to facilitate reliable and easy access. Moreover, LDAS, its interface to the three land models (CLM, NOAH, and VIC), and its input/output modules, will conform ESMF standards, while the output data variables and formats, and the variables passed between LDAS and the three land models, will follow ALMA specification. Finally, LIS also has software components to manage the parallel job processing and monitor hardware status and manage them to ensure sustained high performance output and high availability in the Linux cluster environment. Following is a list of LIS software components:

- Land surface modeling: LDAS and the three land models – CLM, NOAH and VIC. LDAS can be configured to run one, two or all the three land models at the same time.
- Parallel processing: implementation of the parallelization scheme.
- GrADS-DODS server
- Data retrieving
- System monitoring: only applies to the LIS cluster environment.

By the use of modular programming and by conforming to well established standards such as ALMA and ESMF, LIS is expected to provide a flexible, extensible framework to land surface modelers and researchers.

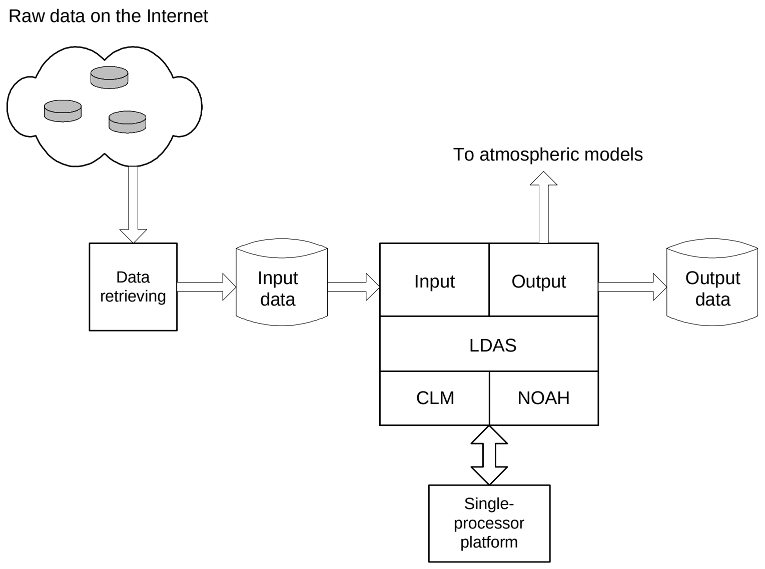


Figure 1: Current Land Data Assimilation System (LDAS) structure. It uses CLM and NOAH land models.

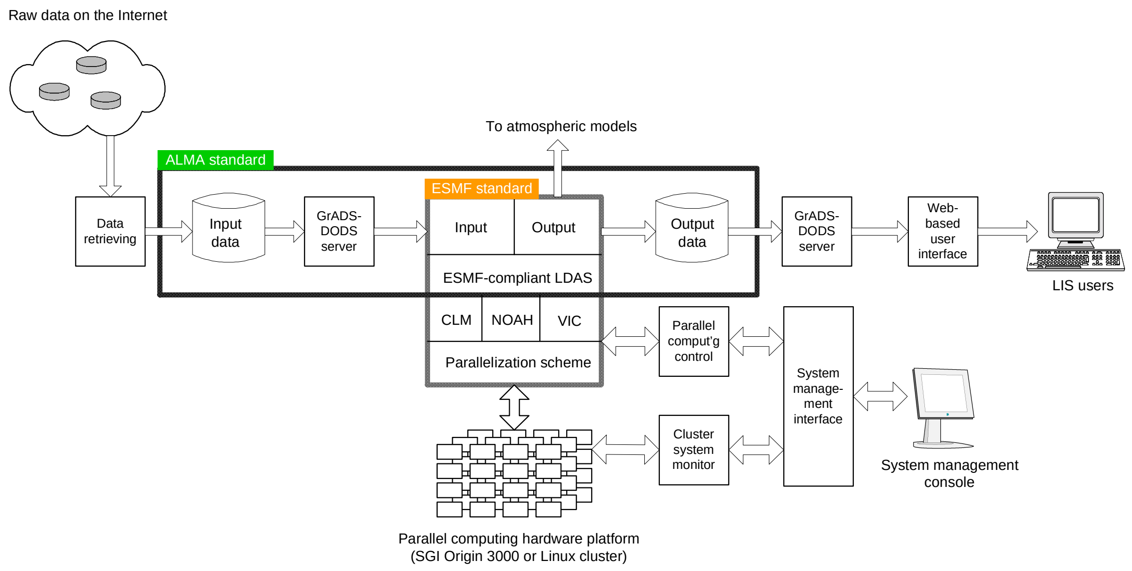


Figure 2: Overview of LIS software architecture and its components designed for LIS cluster. A subset of the components, the LDAS and parallel computing implementation, will also be tested on SGI Origin platforms.

3.1 Software data structures

This section describes the internal software data structures in LIS. As described earlier, the main component that drives different LSMs is LDAS. The one-dimensional land surface models such as CLM, NOAH, and VIC are included as subroutines of LDAS. LDAS, CLM, and NOAH are written in Fortran 90 and these land surface models are interfaced in LDAS through well defined drivers. LDAS code is designed in a modular fashion, with a number of modules used to encapsulate data as well as parameters that are used to solve different governing equations. Please refer to the LDAS code documentation (http://lis.gsfc.nasa.gov/docs/LDAS-Doc/ldas2) for a detailed description of the source code.

A brief description of the modules are presented below:

LDAS Modules
grid_module: This module is an abstract representation of a "grid" used in LDAS. The module includes non model specific parameters such as lat/lon, land/water masks of grid, input/output forcing variables, and variables for temperature assimilation and correction. This module is used by the LDAS main driver and subroutines that are associated with non-model specific computations.

tile_module : This module is a representation of the "tile" described in section 2.1 that is used to simulate sub-grid scale variability. This module includes specification of non-model specific tile variables, such as lat/lon of tile, row/column of tile and properties such as canopy conductance, aerodynamic conductance of the tile.

ldas_module : This module specifies the variables used in LDAS driver such as the model domain specifications, type of land surface model used, type of forcing, specification of source files, etc. It does not include specification of tile space or grid space variables. This module is used by the main driver and subroutines that perform non-model specific computations such as spatial/temporal interpolation.

Driver Modules

The driver modules encapsulate the variables and data types that are involved in the interfacing of land surface models to LDAS. More specifically, they are:

CLM driver modules

drv_module : This module is for the one-dimensional land driver variable specification for CLM. It includes CLM specific parameters such as the driver parameters, timing and diagnostic parameters, etc.

drv_gridmodule : This module is used for the grid space variable specification for CLM. It includes CLM forcing parameters, CLM vegetation parameters, CLM soil parameters, etc.

drv_tilemodule : This module includes tile space variable specification for CLM.

clmtype : This is a module for the one-dimensional CLM variable specification. It includes CLM specific parameters such as water, snow, energy fluxes, soil, vegetation parameters, etc.

NOAH driver modules

noah_module : This module specifies one-dimensional NOAH land driver variable specification. It includes NOAH state parameters, output variables, etc.

VIC structures

VIC includes a number of structures that are used to encapsulate model options, forcing parameters, global simulation parameters, soil and vegetation parameters, etc. The main structures are:

option_struct : This structure is used to store model options.
global_param_struct : This structure is used to store the global parameters defined for the current simulation.
soil_con_struct : This structure is used to store the constant variables for the soil in the current grid cell.
veg_con_struct : This structure is used to store all constant parameters for the vegetation types in the current grid cell.
atmos_data_struct : This structure is used to store the meterological forcing data for each time step.

cell_data_struct : This structure is used to store the grid cell specific variables, not included in the vegetation structures.
energy_bal_struct : This structure is used to store all variables used to compute the energy balance and soil thermal fluxes.
snow_data_struct: This structure is used to store all variables used by the snow accumulation and ablation algorithm, and the snow interception algorithm.

4 Hardware Platforms for LIS

This section describes the hardware operational platforms intended for LIS. The SGI Origin 3000 will be used to implement and demonstrate only the high resolution, parallel, global land surface modeling and data assimilation components (LDAS/CLM/NOAH/VIC) of LIS. The fully operational LIS (with user interfaces and visualization components such as GrADS - DODS) will be demonstrated on a custom designed Linux cluster. The following section describes the hardware design of the cluster.

4.1 LIS cluster architecture

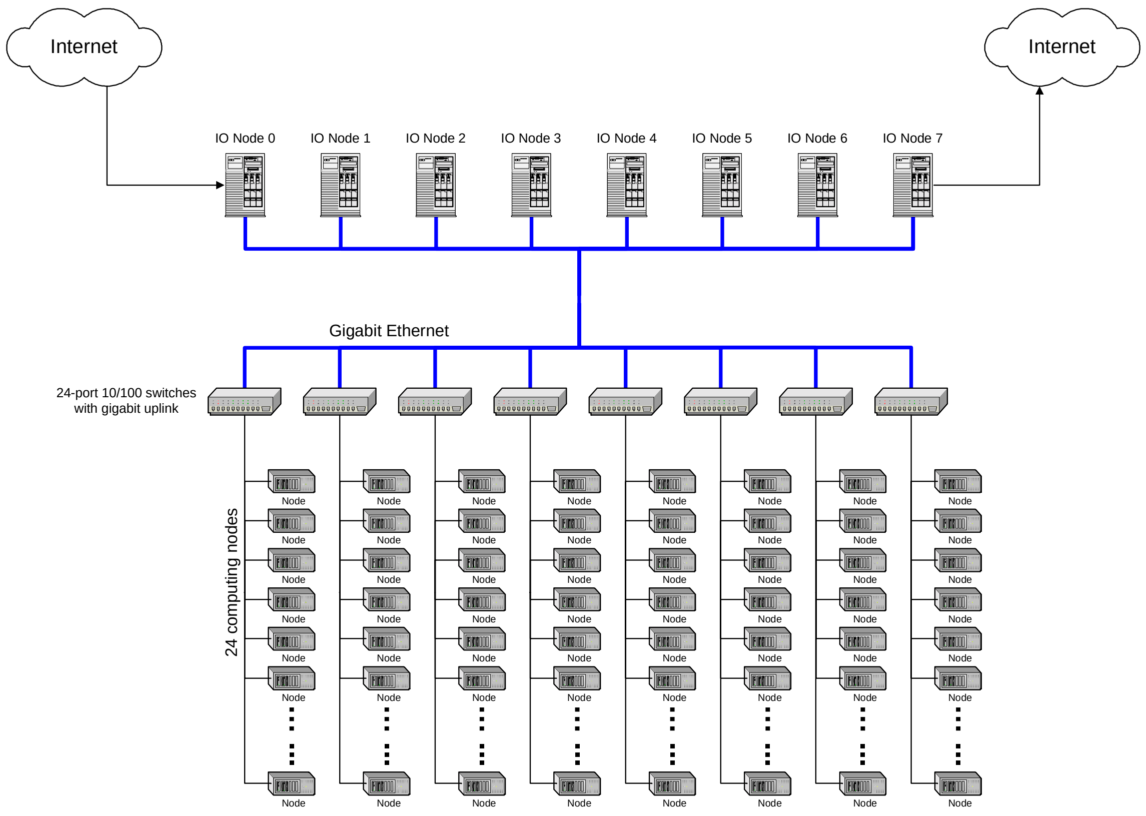


Figure 3: The physical architecture of the LIS Linux cluster. The cluster has 8 IO nodes and 192 compute nodes. Each IO node has dual Athlon CPUs, 2GB RAM and Gigabit NICs, and each compute node has a single Athlon CPU , 512MB RAM and a Fast Ethernet NIC.

Figure 3 shows the physical architecture of the LIS Linux cluster. The cluster consists of 192 computing nodes. The cluster also includes 8 IO (input – output) nodes, specifically to handle the huge data management requirements. These nodes are interconnected with 8 Ethernet switches.

The 192 computing nodes are divided into 8 sub-clusters, with 24 nodes in each sub-cluster, interconnected with fast Ethernet via one of the 8 24-port fast Ethernet switches. Each switch also has two gigabit ports to connect the 8 IO nodes and the other switches.

The use of 8 sub-clusters and 8 IO nodes is mainly for the segregation of network traffic resulting from non-local file IO operations, and for the spreading of data storage so each IO node does not have to deal with single big files. So in average each IO node will only need to serve the IO requests of 24 computing nodes, and only store 1/8 of the output information, which makes the output volume manageable.

4.2 Network traffic estimation within the cluster

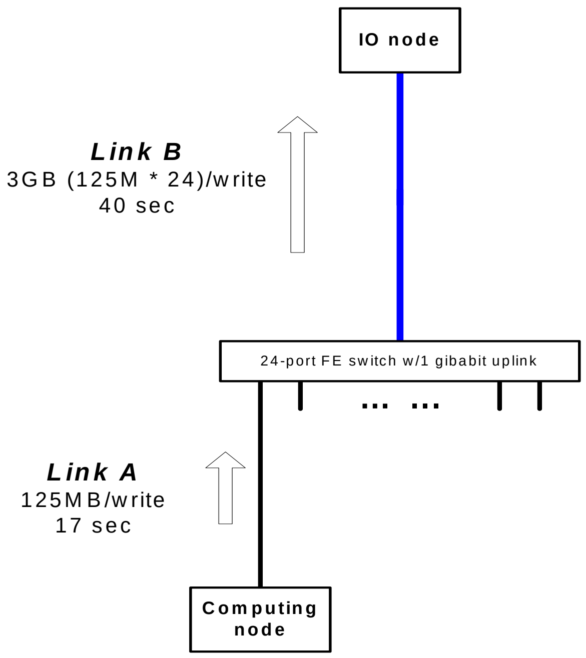


Figure 4: Network traffic estimation within the cluster. The traffic is dominated by the output data flow from the compute nodes to the IO nodes. The output data will go through two network links: Link A, from a compute node to an Ethernet switch; Link B, from the switch's gigabit port to an IO node.

As to be shown in Table 1 below, the total output data volume (200GB/day) produced by the cluster is much larger than the input data volume (279MB/day), so the network traffic is dominated by the upstream traffic from the compute nodes to the IO nodes, where the output data are stored. The data will travel trough two network links: link A -- a compute node to a fast Ethernet switch port; link B -- the switch's gigabit port to an IO node. Figure 4 shows the network traffic between the two network links. Following is the worksheet for the estimation of the traffic at these two links:

Worst case scenario assumption: all the compute nodes are writing the output data to the IO nodes at the same time; effective bandwidth is 60% of the Ethernet wire bandwidth.

Link A traffic:
Data volume each compute node will produce:
200GB/day * (1/192) ~ 1GB/day
Frequency each compute node writes output data to an IO node:
every 3 simulation hours.
Total writes a compute node has:
8 per simulation day
Data volume each write per compute node:
1GB/day * (1/8) = 125MB/write
Time taken for the data to travel over link A:
125MB*8/(100M*60%)= 17 sec

Link B traffic:
In average, the number of compute nodes each IO node receives data from:
24
Total data volume each IO node receives per write:
125MB * 24 = 3GB
Time taken for the data to travel over link B:
3GB*8 /(1G*60%) = 40 sec

In summary, in the worst case scenario, it takes only 57 seconds for the 3-hour simulation data to be transferred from the compute nodes to the IO nodes. In reality, the data traffic will be much spread in time, and the network bandwidth will not be a bottleneck.

5 High performance computing in LIS

Accurate initialization of the land surface moisture, carbon, and energy stores in a fully coupled climate system is critical for meteorological and hydrological prediction. Information about land surface processes is also of critical importance to real-world applications such as agricultural production, water resource management, flood prediction, water supply, etc. The development of LDAS has been motivated by the need for a system that facilitates land surface modeling with an assimilation system to incorporate model derived and remotely sensed data. LDAS system has been successfully used in simulations for North America at 1/8 degree resolution in both real time and long term (50 years) retrospective simulations. However, to truly address the land surface initialization and climate prediction problem, LDAS needs to be implemented globally at high resolution (1km). It can be estimated that the computational and resource requirements increase significantly for global modeling at such high resolutions. The proposed LIS system will aim to make use of scalable computing technologies to meet the challenges posed by the global, high-resolution land surface modeling.

5.1 Parallel processing in land surface modeling

Parallel computing is a powerful programming paradigm to deal with computationally intractable problems. The notion behind parallel programs is to divide the tasks at hand into a number of subtasks and solve them simultaneously using different processors. As a result, a parallel system can improve the performance of the code considerably.

Land surface processes have rather weak horizontal coupling on short time and large space scales, which enables highly efficient scaling across massively parallel computational resources. LIS aims to take advantage of this weak horizontal coupling of land surface processes by using a coarse-grained parallelization scheme, which does not require communication between the compute nodes. This design fits well with the distributed memory nature of the Linux cluster architecture.

The parallel processing code is to break the whole processing job into properly sized small pieces on the IO nodes, and then to distribute the pieces to the compute nodes, to monitor the progress of the small jobs, to maintain balanced loads across the compute nodes, and finally, to collect and assemble the finished pieces and pass the results to the output. The parallel processing component plays a critical role to connect the land surface modeling job to the underlying multi-processor parallel computing hardware platform, in our case, a Linux cluster or an SGI Origin 3000, to achieve the goal of near real-time processing of high-resolution land surface data.

5.2 Land surface modeling in LIS

The land surface modeling component is designed to perform high-performance, parallel simulation of global, regional or local land surface processes with initially three land surface models: the CLM model, the NOAH model and the VIC model. Specifically, the land surface modeling component will interact with the data management components to obtain properly formatted input forcing data, and pass the forcing data, alone with other static parameters, to the three land surface models through the LDAS. Each of the land surface models carries out the simulation on a distributed, parallel hardware platform, either a Linux cluster or a SGI Origin 3000. The results are passed to the output component, which interacts with the data management subsystem to handle the output data. The parallelization process is managed by the system management components. The component provides interface in accordance with ALMA and ESMF standards, wherever applicable.

5.2.1 Structure of land surface modeling component

Figure 5 shows the software structure of the land surface modeling component. The component is designed to be modular with well-defined interfaces that comply with ALMA or ESMF standards. The interface between the land model driver and three land models, CLM, NOAH and VIC, will comply with ESMF and will be general enough so that additional land surface models can be added without much modification of the code. The land surface modeling component is designed in a way that multiple copies can run as different processes in parallel, independent of each other, with each of them processing a different piece of land surface.

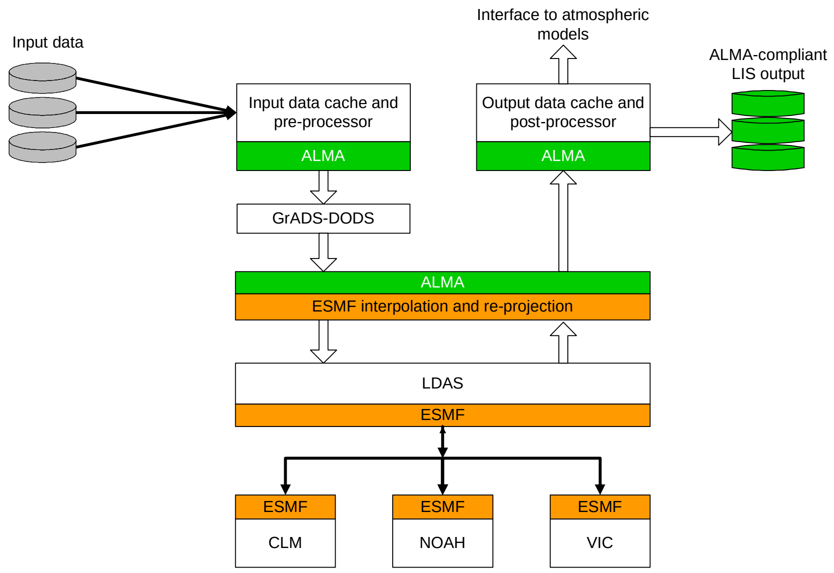

Input data
Interface to atmospheric models
ALMA-compliant LIS output
Input data cache and pre-processor
ALMA
Output data cache and post-processor
ALMA
GrADS-DODS
ALMA
ESMF interpolation and re-projection
LDAS
ESMF
ESMF
CLM
ESMF
NOAH
ESMF
VIC

Figure 5: LIS land surface modeling architecture with ALMA and ESMF interfaces

5.2.2 Implementation of land surface modeling component

The land surface modeling subsystem is designed to be running in parallel, both on a Linux cluster with 200 nodes, and on a SGI Origin 3000 platform with 512 processors. Although the hardware architecture differs greatly between the distributed-memory Linux cluster and the shared-memory SGI Origin 3000, our implementation of the land surface modeling programs will make this architectural difference fairly transparent: On the Linux cluster, each node will run a copy of the land surface modeling process; on the SGI Origin, each CPU will run a copy. Thus we establish a direct correspondence between a node in the Linux cluster and a CPU in the Origin 3000, and the hardware architectural differences will not matter to our design of the software; The land modeling scheme will be able to run on both platforms, with minor modifications in the command line syntax. So in this document whenever we refer to a node in the Linux cluster, it applies equally to a CPU in the Origin 3000.

Interoperability is achieved by following both the ALMA and ESMF standards closely. By following the ALMA standard, the LIS land surface modeling system is guaranteed to exchange data with other land surface modeling systems that are also ALMA-compliant. ESMF standard will allow us to interact with other Earth system models, such as atmospheric models or environmental models with standard interfaces.

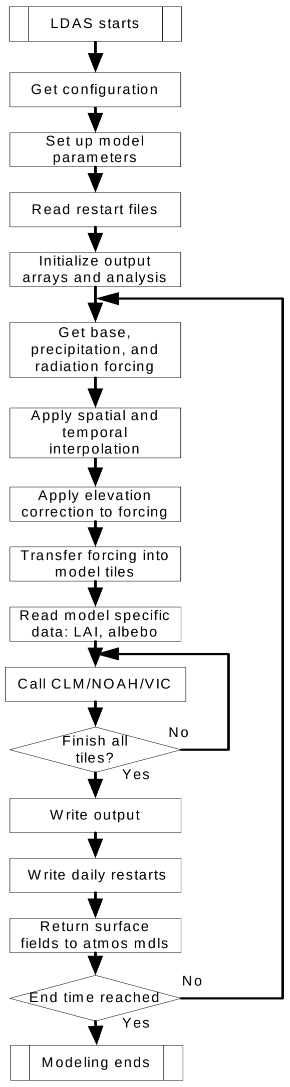

LDAS starts
Get configuration
Set up model parameters
Read restart files
Initialize output arrays and analysis
Get base, precipitation, and radiation forcing
Apply spatial and temporal interpolation
Apply elevation correction to forcing
Transfer forcing into model tiles
Read model specific data: LAI, albebo
Call CLM/NOAH/VIC
Finish all tiles?
No
Yes
Write output
Write daily restarts
Return surface fields to atmos mdls
End time reached
No
Yes
Modeling ends

Figure 6: LIS land surface modeling flowchart.

As shown in Figure 6, and described in detail in the land surface model documentation, land surface models proceed in a manner similar to other physical models. Modeling proceeds given prior knowledge of the spatial and temporal domains of the simulation, in addition to initial conditions and parameters required to solve the equations of water and energy conservation within that domain. Modeling proceeds according to increments of time ("time steps", typically 15 minutes), until the ending time is reached and data is written out for future runs and analysis.

5.3 Compute node job processing

A compute node's job is to run a copy of the land surface modeling subsystem in its process space, compute a piece of land surface obtained from the IO node, and request another piece of land surface from the IO node as soon as it finishes the current piece, until the IO node refuses to give it any pieces, in which case there are no more land pieces are available and the compute node's job is done. Figure 7 shows the flow chart of the compute node's job handling process.

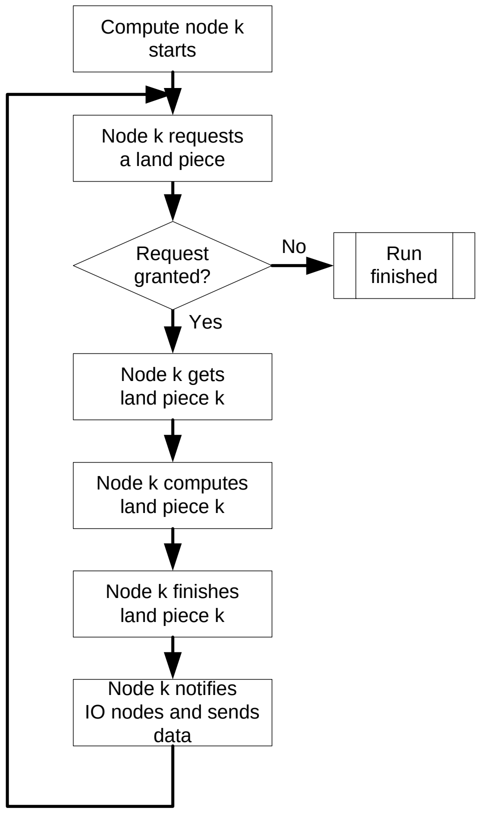

Compute node k starts
Node k requests a land piece
Request granted?
No
Run finished
Yes
Node k gets land piece k
Node k computes land piece k
Node k finishes land piece k
Node k notifies IO nodes and sends data

Figure 7: Compute nodes flowchart for parallel computing of land surface modeling. A compute node does not communicate to other compute nodes.

5.4 IO node job processing

We estimate that at 1km resolution LIS will deal with ~50,000 times more grid points than the 2ºx2.5º resolution LDAS . To satisfy the requirements of real-time operation, the job, which includes a grid representation of the global land surface, must be split into smaller pieces and run in parallel. We plan to divide the global surface into 10,000 small land pieces, and with 1km resolution, each piece would require about 5 times as many computations as the 2ºx2.5º LDAS , and will take a single computing node about 200MB memory to run, and 10 minutes to finish a 1-day simulation, based on the initial performance baselining of LDAS running at both 2ºx2.5º and 0.25ºx0.25º resolutions. The Linux cluster can consume approximately 200 pieces per round, and under ideal conditions, it will take the whole cluster about 50 rounds to finish the whole job. This will take 500 minutes, or about 9 hours, to finish a 1-day simulation of the whole global land surface, which satisfies the real-time requirement with enough extra room. We expect that the timings on the SGI Origin will be comparable to those on the cluster, although memory and disk limitations, some imposed by the queue structure, will likely prohibit effective use of that system for demonstrating LIS in a near-real-time mode. However, we plan to demonstrate the LIS on the SGI Origin system as proof-of-concept.

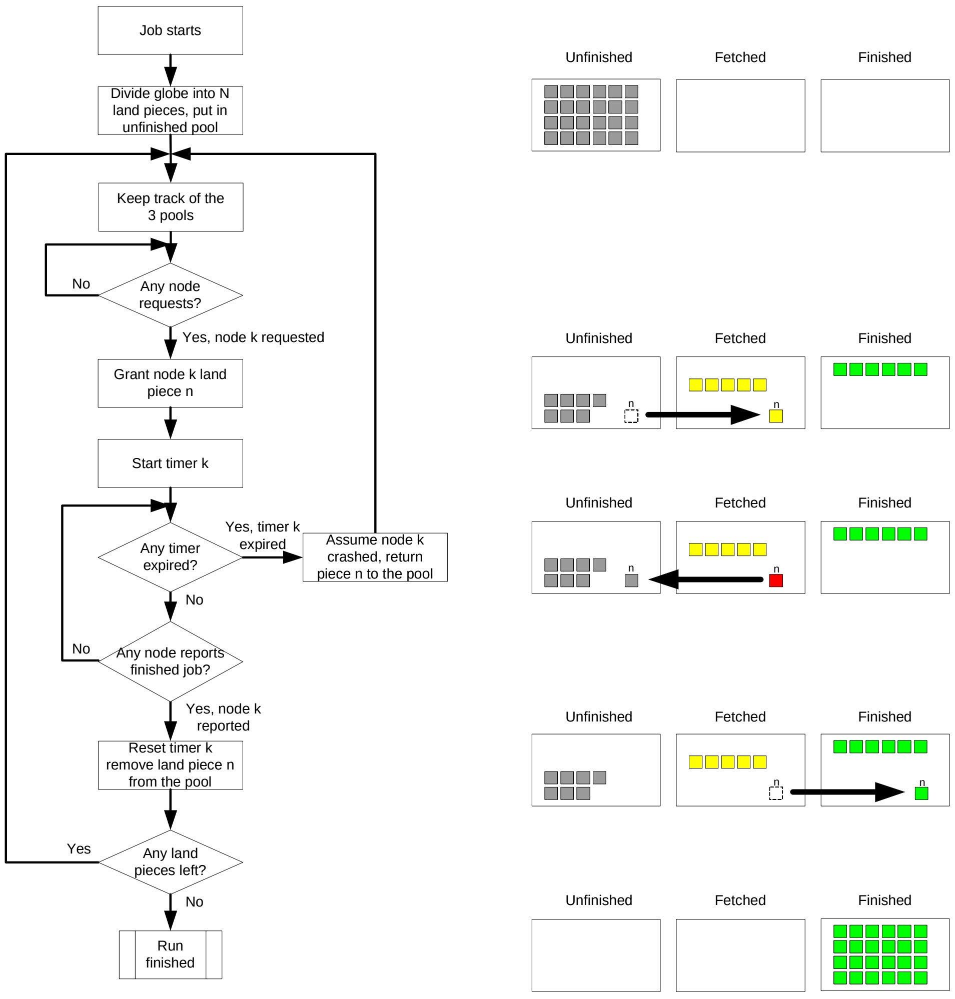

Job starts
Divide globe into N land pieces, put in unfinished pool
Keep track of the 3 pools
No
Any node requests?
Yes, node k requested
Grant node k land piece n
Start timer k
Any timer expired?
Yes, timer k expired
Assume node k crashed, return piece n to the pool
No
No
Any node reports finished job?
Yes, node k reported
Reset timer k remove land piece n from the pool
Yes
Any land pieces left?
No
Run finished
Unfinished
Fetched
Finished
n

Figure 8: Parallel computing control flowchart (left) and parallelization scheme (right) of a master node.

We propose to use a modified version of the "pool of tasks" scheme for the parallel processing of the land pieces. A pool of tasks paradigm is equivalent to a master – slave programming notion, where a single processor will act as a master node and distribute jobs to the slave (compute) nodes. In the LIS "pool of tasks" design, one of the IO nodes will act as a master node and another IO node will be designated as a backup to it. The master node will keep three tables on hand when starting the job: table of unfinished-jobs, finished-jobs, and jobs-fetched. At the beginning, the 10,000 land pieces are listed in the "unfinished" table, and each compute node comes to the master to fetch a piece from it, and starts working on it. The master node then moves the fetched jobs to the "jobs-fetched" table, and starts a timer for each fetched job. The timer specification will be based on the existing knowledge of a single execution of a land surface model. When a compute node finishes a job and notifies the master node before the job's corresponding timer runs out, this piece is regarded a finished job, and the master node moves it from the "fetched" table to the "finished" table. And the compute node goes on to fetch another job until the "unfinished" table is empty. If a fetched job's timer runs out before the compute node reports back, the master node then assumes that that particular compute node must have crashed, and then moves that timed-out job from the "fetched" table back to the "unfinished" table for other compute nodes to fetch. Figure 7 shows the flowchart (left) of the master node’s job handling and scheduling process, and the various status of the three tables (right) the master node uses to keep track of the job progress at different corresponding stages in the flowchart.

To maximize throughput of the system in a parallel environment, load balancing is required to keep the compute nodes busy by efficiently distributing the workload. The use of a "pool of tasks" is effective in achieving automatic load balancing by minimizing the idle times of compute nodes, since the nodes that finish their computations will request more tasks than the ones that require more time for their computations. This automatic, asynchronous scheduling help in keeping the compute nodes busy without having to wait for other node's computations.

As shown in Figure 8, as the land surface modeling process starts, the master node divides the globe into a number of smaller pieces. The inputs required by the land surface models, namely, initial conditions, boundary conditions, and parameters will be provided to the compute nodes before the land surface model run begins. The modeling process can be a fresh initialization (cold start) or a restart from a previously finished run. This process also requires preprocessing of the data such as time/space interpolation. The output from each compute node, after the computation, will be reassembled at the IO nodes.

6 Data Management in LIS

The data management subsystem in LIS is composed of the following functions: input data retrieval from the Internet, data pre-processing and post-processing, data interpolation and sub-setting, output data aggregation, storage, backup and retrieval. It links the other subsystem together, and ensures smooth end-to-end data flow, from the input raw data all the way to the output data satisfying LIS users’ various requests. The following sections describe the data flow and volume used in LIS operation, the use of GrADS-DODS server for data management, visualization, etc., and other functions such as data retrieval.

6.1 Data flow and volume in LIS

Figure 9 shows the global logical data flow of LIS system on the LIS cluster platform. On SGI Origin platform, the IO nodes in Figure 3 will be replaced by local disks for the IO functions, and the compute nodes are replaced with the same number of CPUs. Input data will be pre-staged on SGI instead of using GrADS-DODS servers.

LIS will deal with three categories of global data: parameter data, input forcing data and output data. At the top level of the system design, the global data are represented by data files of various formats.

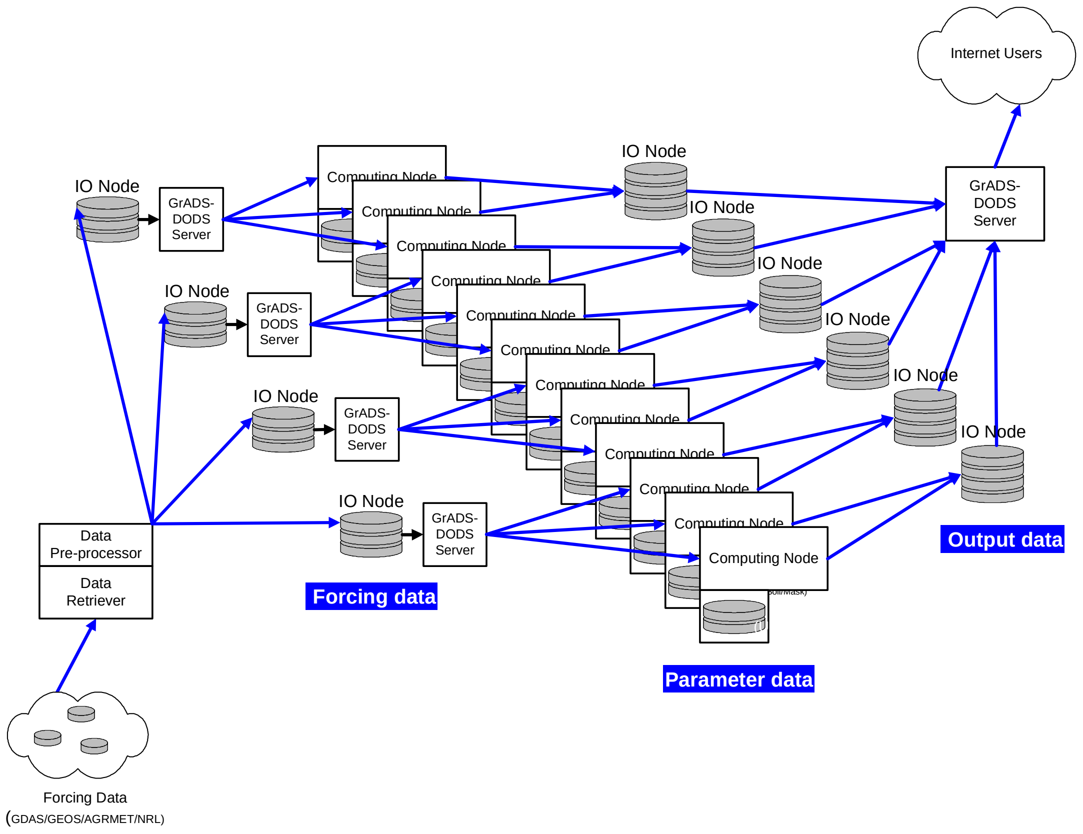


Figure 9: LIS global logical data flow on LIS Linux cluster. Physically, the IO nodes for input data and output data on the two sides of the cluster are the same IO node computers. On SGI, the flow is similar, except the IO nodes will be replaced by local hard disks, and the compute nodes will be replaced by CPUs. GrADS-DODS servers will not be used on SGI. Instead, the data will be pre-staged.

The parameter data include the vegetation classification, land mask, etc., with a size of about 136 GB. Since these data will not be updated frequently, we will put a copy of these data on each compute node's local disk to reduce network traffic. Currently the bulk of the data are saved as ASCII data, and we will convert the data into binary format to allow all the static data to fit on the node's 80 GB disk.

The forcing data, fetched from various locations on the Internet, need to be fed to the compute nodes at regular intervals. The total traffic is estimated to be 279 MB/day, which is not significant compared to the output data traffic. We designate one of the IO nodes to fetch and pre-process the data, then send a copy of the forcing data to the other IO nodes via NFS system. When a compute node needs the forcing data, it will contact the IO node, which corresponds to the sub-cluster it belongs to without bothering other IO nodes. To further reduce the IO network traffic, each IO node will run the GrADS-DODS server to feed the compute nodes with the sub-set of the data they need.

The output data will be stored on the IO nodes too, and served to users via a GrADS-DODS server running on one of the IO nodes. Since it is not feasible to store the output in a single file (200 GB/day), we want to distribute the data across all the IO nodes. To keep the huge output data volume manageable, we designed a storage scheme that will distribute the land surface variables in the output data across the IO nodes. Since there are 40-48 variables in the output data, with some of them having multiple levels, we can let each IO node to store the global data of only 6 or so of the output variables. So on average, the I/O traffic is segregated and each IO node is only taking 1/8 of the total data traffic, and the subsequent operations by the GrADS-DODS servers are greatly simplified.

Table 1 lists all the global data files and specifications. As described in Section 2, these files specify the parameters, initial and boundary conditions required for the land surface model runs. For e.g, the forcing data translates to variables such as total precipitation, convective precipitation, downward shortwave and longwave radiation, near surface air temperature, near surface specific humidity, near surface U, V, winds and surface pressure. In addition to these files, the user also specifies parameters such as the spatial and temporal resolution, the land surface model, etc. LDAS also allows the user to initialize state variables, either by specifying a global uniform value or taken from a restart file produced by a prior run. Please refer to the LDAS source code documentation (http://lis.gsfc.nasa.gov/docs/LDAS-Doc/ldas2/) for a detailed description of the input/output routines corresponding to each file. The output from the land models translates to variables such as soil moisture, surface runoffs, canopy transpiration, etc. A list of the LDAS variables passed between the modules, following the ALMA convention, is presented in Appendix A.

Table 1: LIS global data files

LIS data files and estimated data volume for LIS with 1km x 1km resolution, based on the data used for LDAS ¼ x ¼

Dataset	Information	Desired Resolution	Native format	Apprx. Size	Update frequency
UMD Vegetation classification map	This file lists the frequency with which of each of the 14 vegetation types occurs in each of the ¼ degree LDAS grid boxes. See http://ldas.gsfc.nasa.gov/GLDAS/VEG/GLDASveg.shtml for a detailed description.	1km x 1km	ASCII	65G	Static
UMD Land mask	This ascii file contains the LDAS unified land/sea mask. See http://ldas.gsfc.nasa.gov/GLDAS/VEG/GLDASveg.shtml for a detailed description	1km x 1km	ASCII	18G	Static
Soil classification map	The soil parameter maps used in LDAS were derived from the global soils dataset of Reynolds et al. (1999). That dataset includes the percentages of sand, silt, and clay, among other fields, and is based on the United Nations Food and Agriculture Organization (FAO) Soil Map of the World linked to a global database of over 1300 soil pedons. The LDAS soil color map was interpolated from a 2 x 2.5 degree global map produced by NCAR	1km x 1km	Binary	20G	Static
Soil color map		1km x 1km	Binary	2G	Static
Sand fraction file		1km x 1km	Binary	6G	Static
Clay fraction map		1km x 1km	Binary	6G	Static
Leaf Area Index (LAI)	This was generated using three information sources: (1) an 8km resolution time series of LAI, which was derived by scientists at Boston University (Myneni et al. 1997) from AVHRR measurements of normalized difference	1km x 1km	Binary	1M	Static

AVHRR – derived LAI climatology	vegetation index (NDVI) and other satellite observations. (2) A climatology based on the 8km time series and (3) the 1km UMD vegetation type classification.	1km x 1km	Binary	5G	Static
Static file size				**125G**	
GEOS forcing data	Obtained from GSFC's Goddard Earth Observing System Data Assimilation System (GEOS) (Pfaendtner et al. 1995) version 4.3 that supports level-4 product generation for the NASA Terra satellite (Atlas and Lucchesi 2000).	1 deg	Binary	25M/day	Every 3 hours
GDAS forcing data	The Global Data Assimilation System (GDAS) is the global, operational weather forecast model of NCEP (Derber et al 1991). LDAS makes use of GDAS 0, 0.3, and, as needed, 6 (hour) forecasts, which are produced at 6 hour intervals	Native T170 ~0.7 deg	GRIB	50M/day	Every 6 hours
AGRMET SW flux data	LDAS estimates global, downward shortwave and longwave radiation fluxes using a procedure from the Air Force Weather Agency's (AFWA) Agricultural Meteorology modeling system (AGRMET). It utilizes the AFWA Real Time Nephanalysis (RTNEPH) 3-hourly cloud maps (Hamill et al. 1992), and the AFWA daily snow depth (SNODEP) maps (Kopp and Kiess 1996) to calculate surface downwelling shortwave radiation using the algorithms of Shapiro (1987)	~ 48 km	Binary	48M/day	Every 1 hour
AGRMET LW flux data			Binary	144M/day	Every 1 hour
NRL precipitation data	Near-real time satellite-derived precipitation data is obtained from the U. S. Naval Research Laboratory (NRL). NRL produces precipitation fields based on both geostationary satellite infrared (IR) cloud top temperature measurements and microwave observation techniques (Turk et al. 2000)	¼ degree	Binary	12M/day	Every 6 hours
Total data input flux				**279M/day**	
Estimated output volume for LIS (1/100 x 1/100) Resolution, based on GLDAS runs with 2x2.5 resolution. Data output interval is assumed to be the same					
CLM output data	Output from LDAS runs	1km x 1km	GRIB	200G/day	Every 15 min
Total data output flux				**200G/day**	

6.2 GrADS-DODS server structure

GrADS-DODS servers will be employed both to serve the input data to the land surface computing code, and to serve the output to the Internet users. Figure 10 shows the architecture of the GrADS-DODS server. A GrADS-DODS server uses a typical client-server architecture to communicate with the DODS clients. The communication protocol between a client and a server is HTTP. A GrADS-DODS server has the following components: Java servlets contained in the Tomcat servlet container, to handle the client requests and server replies via HTTP protocol; DODS server APIs, to parse the DODS requests and package output data; interface code, to translate the DODS requests into GrADS calls; and finally, GrADS running in batch mode, to actually process the requests, and perform data-retrieving, sub-setting and processing on the server side.

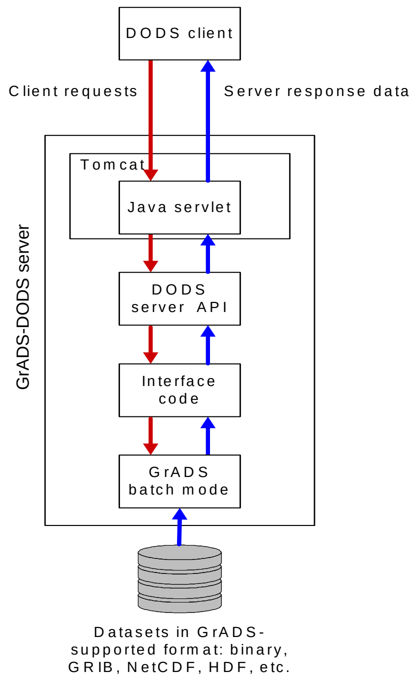


Figure 10: GrADS-DODS server architecture.

6.3 Description for data retrieving component

The data retrieving component locates and downloads various atmospheric forcing data sets, as specified in Table 1, at regular intervals, from the Internet to LIS's local disks. The data retrieving component will also perform some basic pre-processing on the forcing data.

6.3.1 Implementation

The data retrieving component is implemented as a multi-process structure, with each process dealing with a specific data set, so in case a data set takes unusually long time, it will not block the other processes' progress. Following is the pseudo-code of the data retrieving component:

```
Define data sources:
     DS[1]:     URL1
     DS[2]:     URL2
     …
     DS[n]:     URLn
End Define data sources

For I=1, n Do
   Start process(I) (non-blocking start)
End For

Define Process (I)
    Fetch data from DS[I];
    Pre-process DS[I];
    Return;
End Define Process (I)
```

7 Description for system monitoring component

The system monitoring component is responsible for monitoring, maintaining and administering the LIS system on the Linux cluster to ensure its reliable operation and optimal performance output.

We categorize the system management function into four levels: hardware level, interconnect level, operating system level and application software level. For the SGI Origin 3000 platform, we are not involved in the management of the hardware and interconnect levels. But for the Linux cluster, the hardware and interconnect level management is our responsibility and is critical to the overall stability and performance of the LIS system.

The hardware level system management involves power-up and shutdown of the nodes, booting strategy and hardware status monitoring. Interconnect level management requires the monitoring of the link status of the network nodes, bandwidth usage and traffic statistics. Operating system level management takes care of system resource usages, such as CPU, memory and disk space usage. Application level management oversees the progress of the LIS jobs, configures different runs, analyze performance bottlenecks, and obtain performance profiles for fine-tuning. Dynamic error and diagnostic logs will be maintained for LDAS and the land surface models during the operation of LIS. The diagnostic logs will be available to the end users.

7.1 Hardware monitoring data

The following table summarizes the system data of various levels the management subsystem is designed to collect and analyze.

Table 2: Hardware monitoring and management data collection

LIS Cluster System Monitoring and Management Data		
Category	***Data Items***	***Update frequency***
Application level	Overall cpu/mem of each process	1min
	Overall progress of whole job	2min
	Progress of each process	1min
	Timing of each module	sampled, off-line
	Memory usage of each module	sampled, off-line
Operating system level	Total memory usage & biggest user	2min
	Total CPU usage & biggest user	2min
	Total disk space usage	2min
	System up-time and running procs	2min
Interconnect level	Bandwidth usage of each node	2min
	Bandwidth usage of switches	2min
	Latency measurements	2min
	Packet drops measurements	2min
Hardware level	Fan speeds	10min
	Chasis temperature	10min
	Power supplies voltage	10min

7.2 Architecture and implementation

The variety of system variables and management duties requires us to design a management system with modules performing individual and well-defined tasks. Figure 11 shows the structured design of the system management functionalities for the Linux cluster platform. We will not implement such a system on SGI because the SGI platform is not under our management control.

On the hardware level, we will design scripts to take advantage of the “Wake-on-Lan” technology for powering up the nodes smoothly in a well-defined pattern. The nodes will be able to boot across the network with the PXE technology, as well as from the local disk, to centralize system software management. After booting, each node’s hardware parameters, such as CPU temperature, cooling fan speeds and power supply voltages, will be collected by kernel modules called “lm-sensors”, and sent to the central management station with web-based display with automatic updates.

On the interconnect level, we will use SNMP protocol as the underlying data collection and management mechanism, interfaced with MRTG for web-based display of network statistics. Additional network data can also be collected by Big Brother system and network monitor, also with web output.

On the operating system level, we will use SNMP and various OS shell commands and utilities to collect system data, and use MRTG and Big Brother as the interface.

On the application level, we will develop CGI scripts, interfaced with OS commands and utilities, to provide a web-interface for the monitoring and control of LIS jobs and processes. Standard performance profiling and debugging tools will be used off-line to analyze sample runs for trouble-shooting and performance fine-tuning.

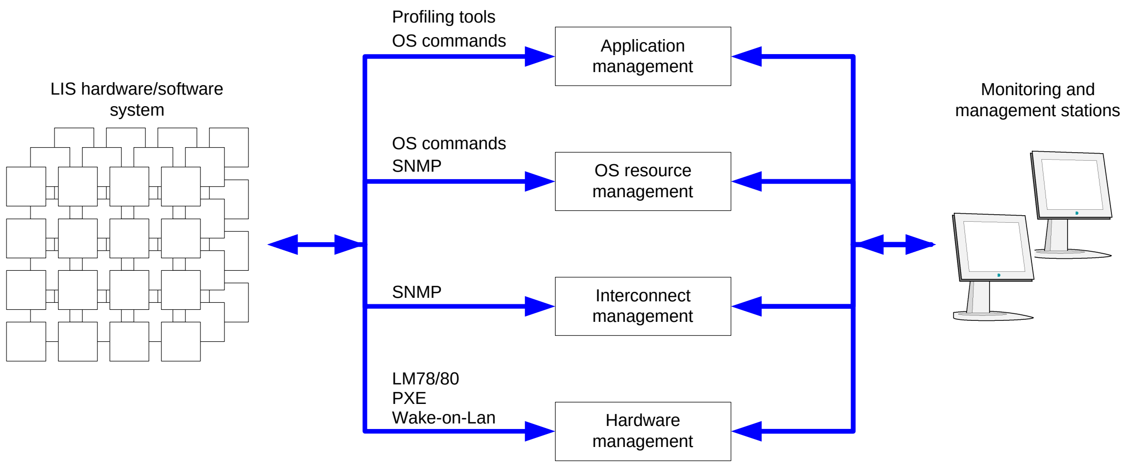


Figure 11: LIS system monitoring and management architecture for the LIS Linux cluster. This system will not be implemented on SGI since it is not under our control.

8 User interface design

The user interface in LIS is an important component of LIS that will allow the interactive, flexible, use of the LIS hardware and software to users. The LIS user interface is intended to be web-based, and designed to allow for cascading complexity depending on the level of user's need to control the system. The following sections present various facets of the user interface design of LIS.

8.1 User interface components

The user interface subsystem takes a typical multi-tier client-server system architecture. On the client side, a user has three types of client programs to use as the front-end: a web browser, a ftp client program (which can be integrated in a web browser), or a DODS client program. On the server side, a general purpose web server will be used to serve clients with a web browser, and a GrADS-DODS server will be deployed to serve DODS clients, and a FTP server to server ftp clients. Besides theses components, CGI scripts and CGI-GrADS gateway scripts will be used as the middleware to perform dynamic processing based on users' interactive requests sent through web browsers. Figure 12 shows the user interface architecture design.

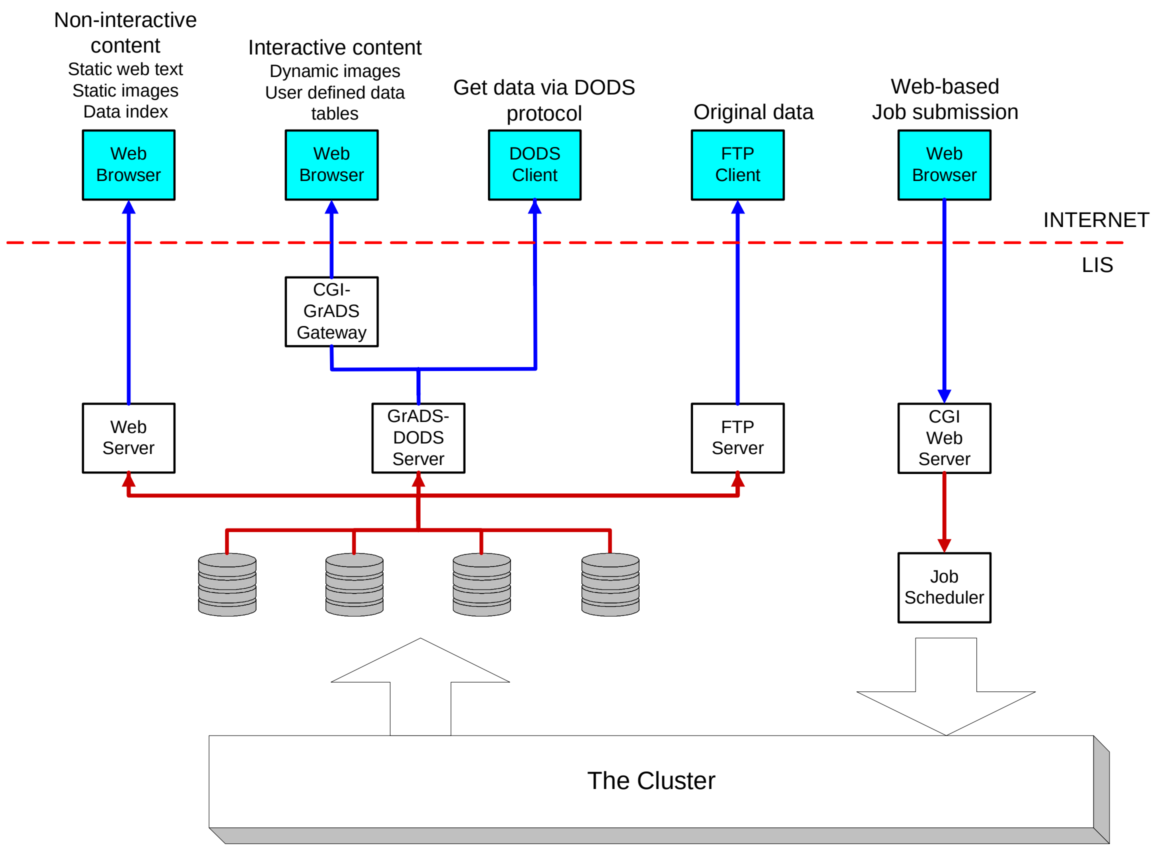


Figure 12: LIS user interface architecture.

8.2 Sample prototype interface

Figure 13 shows representative examples of the web-base interface objects to be implemented in the user interface. Text boxes ask for a user's free input, which are mostly used in the user authentication process. Check boxes prompt a user to make one or more choices (Radio buttons ask for one choice). Drop-down menus limit a user's options to predefined ones, which are suitable for model parameter input. Check boxes and text boxes can be combined to provide a user with selectable search criteria, for example. Finally, graphics are either pre-produced or produced as results of a user's data query or data analysis. Figure 14 is a screenshot of the LIS entry page.

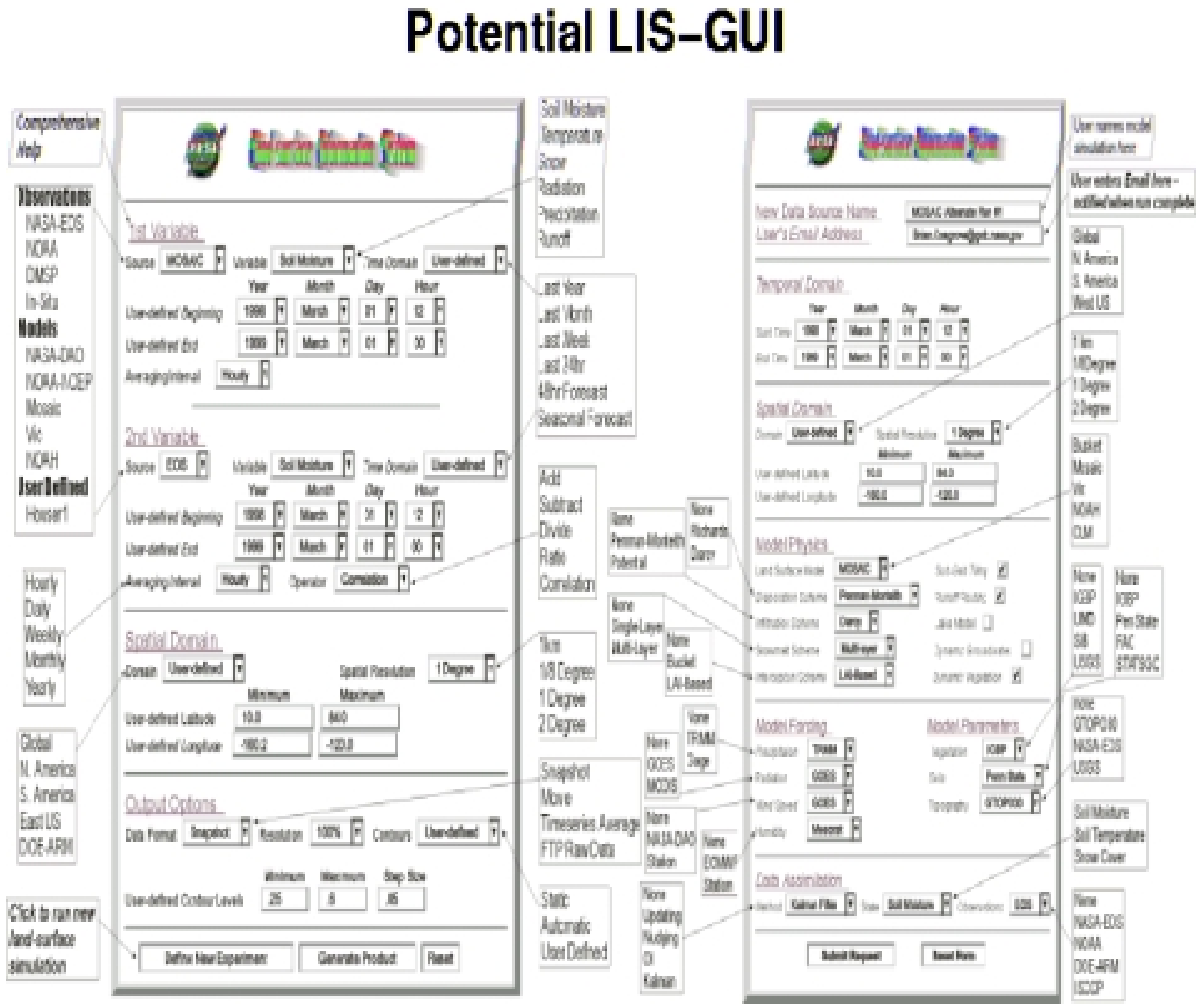
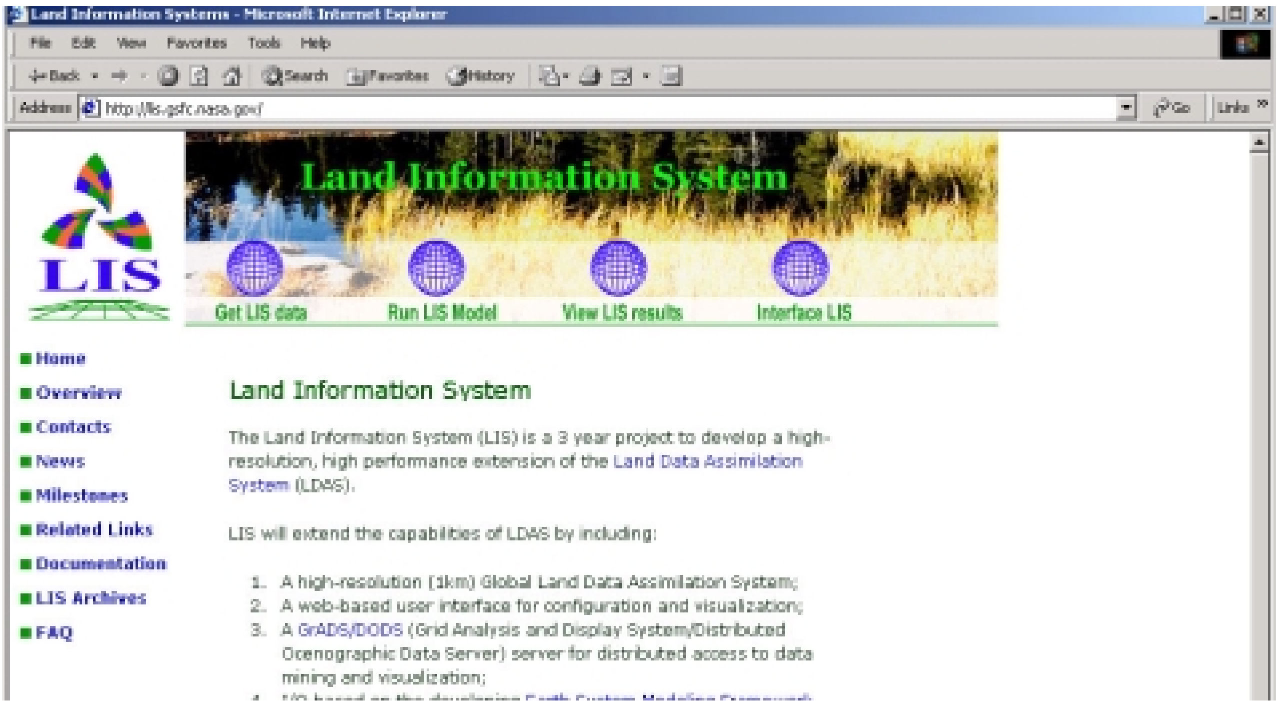


Figure 13: Sample of web-based user interface objects.


Land Information Systems - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://lis.gsfc.nasa.gov/
Go Links
LIS
Get LIS data
Run LIS Model
View LIS results
Interface LIS
Home
Overview
Contacts
News
Milestones
Related Links
Documentation
LIS Archives
FAQ

Land Information System

The Land Information System (LIS) is a 3 year project to develop a high-resolution, high performance extension of the Land Data Assimilation System (LDAS).

LIS will extend the capabilities of LDAS by including:

1. A high-resolution (1km) Global Land Data Assimilation System;
2. A web-based user interface for configuration and visualization;
3. A GrADS/DODS (Grid Analysis and Display System/Distributed Ocenographic Data Server) server for distributed access to data mining and visualization;

Figure 14: Screenshot of LIS web entry page.

8.3 User levels and security design

Outside users accessing the LIS are categorized into three levels, associated with different levels of data access and security requirements.

Level 1 users are the general public, who will access the LIS data primarily through a standard web browser. Information provided to this class includes static images and text, and some limited interactive content such as GIF/JPG/PNG images generated on the fly in response to users' regulated web input. The static content, most of which is static html pages, is served via the web server, while the interactive content is generated via a three-tier architecture with server-side GrADS as the image engine and below it the GrADS-DODS server as the data engine to feed the server-side GrADS. This group of users does not have direct access to the data or LIS scientific computing power system, and their usage of system resources is very limited. Therefore, for this class of users we do not enforce any additional authentication or authorization procedures. It is also our intention to facilitate easy access to the data for education and outreach purposes.

Level 2 users have direct access to LIS data, either through our GrADS-DODS server by using a DODS client, or directly through ftp fetches. The GrADS-DODS server provides the users with the ability and flexibility to get only a sub-set of the data they need. To be authorized as Level 2 users, they will have to register with us first by filling out web forms, and they will be authenticated using password and source IP addresses before accessing the data. The GrADS-DODS server will impose a limit on system resource usages. The GrADS-DODS server allows the system administrator to limit the system usage by configuring the following parameters for each authorized IP address:

Table 3: Configurable GrADS-DODS parameters for access to level 2 users of LIS

Parameter	Description
Subset limit	Sets the maximum size in megabytes of a subset
Generate limit	Sets the maximum size in kilobytes of a generated dataset
Upload limit	Sets the maximum size in kilobytes of an uploaded dataset
Time limit	Sets the maximum time in milliseconds that a dataset generation task is allowed
Hit limit	Sets the maximum number of hits per hour permitted from a specific IP
Abuse limit	Sets that length of time in hours an IP address will be blocked out after exceeding the hit limit
Deny datasets	A comma delimited list of datasets that should not be accessible
Allow datasets	A comma delimited list of datasets that should be accessible

Level 3 users can configure LIS model runs to their taste using the web interface. The configuration parameters they enter in the web form will be converted to LIS configuration files to control model runs. All the parameters will have default values.

The configuration parameters needed for a LIS model run fall into either temporal domain, spatial domain, model physics, model forcing, or output. The temporal domain parameters are the start and end time. The spatial domain parameters needed are the area over which the model will be run (local, regional, global, or user defined in latitude/longitude), and spatial resolution (1 degree, 1/4 degree, or 1 km). The model physics parameters needed are the Land Surface Model, the evaporation scheme, the snowmelt scheme, the leaf index scheme, subgrid tiling on/off, runoff routing on/off and dynamic vegetation on/off. The model forcing configuration parameters needed are the precipitation input file source, the radiation input file source, the wind speed input file source, the humidity input file source, the temperature input file source, the vegetation input file source, the soil input file source, and the topography input file source. The output configuration parameters define the output data format and resolution.

A representative sample listing of the required parameters of the LIS model run is included in Appendix B.

References

ALMA: http://www.lmd.jussieu.fr/ALMA/

Atlas, R. M., and R. Lucchesi, File Specification for GEOS-DAS Gridded Output. Available online at : http://dao.gsfc.nasa.gov/DAO_docs/File_Spec_v4.3.html , 2000.

Chen, F., K. Mitchell, J. Schaake, Y. Xue, H. Pan, V. Koren, Y. Duan, M. Ek, and A. Betts, "Modeling of land-surface evaporation by four schemes and comparison with FIFE observations", *J. Geophys. Res.*, 101, D3, 7251-7268, 1996.

CLM: http://www.cgd.ucar.edu/tss/clm/

Collatz G. J., C. Grivet, J. T. Ball, and J. A. Berry, J. A. "Physiological and Environmental Regulation of Stomatal Conductance: Photosynthesis and Transpiration: A Model that includes a Laminar Boundary Layer", *Agric. For. Meteorol.* , 5, pp 107 -- 136, 1991.

Derber, J. C., D. F. Parrish, and S. J. Lord, "The new global operational analysis system at the National Meteorological Center", *Wea. And Forecasting,* 6, pp 538-547, 1991.

ESMF: http://www.esmf.ucar.edu/

GrADS-DODS server: http://grads.iges.org/grads/gds/

Hamill, T. M., R. P. d'Entremont, and J. T. Bunting, "A description of the Air Force real-time nephanalysis model", *Wea. Forecasting,* 7, pp 238-306, 1992.

Hofstee, H. P., J. J. Likkien, and J. L. A. Van De Snepscheut "A Distributed Implementation of a Task Pool". *Research Directions in High-Level Parallel Programming Languages*, pp 338--348, 1991.

Jarvis, P. G., " The interpretation of leaf water potential and stomatal conductance found in canopies in the field", *Phil. Trans. R. Soc.* London, Ser. B, 273, pp 593 – 610, 1976.

Kopp, T. J. and R. B. Kiess, "The Air Force Global Weather Central cloud analysis model", *AMS 15th Conf. on Weather Analysis and Forecasting*, Norfolk, VA, pp 220-222, 1996.

LDAS: http://ldas.gsfc.nasa.gov/

NOAH: http://www.emc.ncep.noaa.gov/mmb/gcp/noahlsm/README_2.2.htm

Pfaendtner, J., S. Bloom, D. Lamich, M. Seablom, M. Sienkiewicz, J. Stobbie, and A. da Silva, "Documentation of the Goddard Earth Observing System (GEOS) Data

Assimilation System – Version 1”, *NASA Technical Memorandum* 104606, 4, pp 44, 1995.

Reynolds, C. A., T. J. Jackson, and W. J. Rawls, “Estimating available water content by linking the FAO Soil Map of the World with global soil profile databases and pedo-transfer functions” *American Geophysical Union, Fall Meeting, Eos Trans. AGU,* 80, 1999.

Richards, L. A., “Capillary conduction of liquids in porous media”, *Physics*, 1, pp 318—333, 1931.

Rogers, E., T. L. Black, D. G. Deaven, G. J. DiMego, Q. Zhao, M. Baldwin, N. W. Junker, and Y. Lin, “Changes to the operational "early" eta analysis / forecast system at the National Centers for Environmental Prediction” *Wea. Forecasting*, 11, pp 391-413, 1996.

Shapiro, R. “A simple model for the calculation of the flux of direct and diffuse solar radiation through the atmosphere”, AFGL-TR-87-0200, Air Force Geophysics Lab, Hanscom AFB, MA.

Turk, F. J., G. Rohaly, J. D. Hawkins, E. A. Smith, A. Grose, F. S. Marzano, A. Mugnai, and V. Levizzani, “Analysis and assimilation of rainfall from blended SSM/I, TRMM, and geostationary satellite data”, *AMS 10th Conf. On Sat. Meteor. and Ocean.*, Long Beach, CA, 9-14 January, pp 66-69, 2000.

VIC: http://www.hydro.washington.edu/Lettenmaier/Models/VIC/VIChome.html

Appendix A

List of LDAS variables passed between the modules (Following ALMA Convention)

nswrs	Net Surface Shortwave Radiation (W/m2)
nlwrs	Net Surface Longwave Radiation (W/m2)
lhtfl	Latent Heat Flux (W/m2)
shtfl	Sensible Heat Flux (W/m2)
gflux	Ground Heat Flux (W/m2)
snohf	Snow Phase Change Heat Flux (W/m2)
dswrf	Downward Surface Shortwave Radiation (W/m2)
dlwrf	Downward Surface Longwave Radiation (W/m2)
asnow	Snowfall (kg/m2)
arain	Rainfall (kg/m2)
evp	Total Evaporation (kg/m2)
ssrun	Surface Runoff (kg/m2)
bgrun	Subsurface Runoff (kg/m2)
snom	Snowmelt (kg/m2)
snowt	Snow Temperature (K)
vegt	Canopy Temperature (K)
baret	Bare Soil Surface Temperature (K)
avsft	Average Surface Temperature (K)
radt	Effective Radiative Surface Temperature (K)
albdo	Surface Albedo, All Wavelengths (%)
weasd	Snowpack Water Equivalent (kg/m2)
cwat	Plant Canopy Surface Water Storage (kg/m2)
soilmc	Total Column Soil Moisture (kg/m2)
soilmr	Root Zone Soil Moisture (kg/m2)
soilmt1	Top 1-meter Soil Moisture (kg/m2)
mstavc	Total Soil Column Wetness (%)
mstavr	Root Zone Wetness (%)
evcw	Canopy Surface Water Evaporation (W/m2)
trans	Canopy Transpiration (W/m2)
evbs	Bare Soil Evaporation (W/m2)
sbsno	Snow Evaporation (W/m2)
pevpr	Potential Evaporation (W/m2)
acond	Aerodynamic Conductance (m/s)
lai	Leaf Area Index
snod	Snow Depth (m)
snoc	Snow Cover (%)
salbd	Snow Albedo (%)
tmp2m	Two Meter Temperature (K)
humid	Two Meter Humidity (kg/kg)
uwind	Ten Meter U Wind (m/s)
vwind	Ten Meter V Wind (m/s)
sfcprs	Surface Pressure (mb)
soilt	Soil Temperature (K)
soilm	Soil Moisture (kg/m2)
lsoil	Liquid Soil Moisture (kg/m2)

Appendix B

Sample input card file for the execution of LIS model run

```
!===> Fundamental parameters necessary for running LDAS
&driver
LDAS%DOMAIN = 1 ! Model domain (1=NLDAS,2=GLDAS1/4,3=GLDAS2x2_5)
LDAS%LSM    = 1 ! Land surface model (2=CLM,4=NOAH)
LDAS%FORCE  = 3 ! Forcing data type (1=GDAS,2=GEOS,3=ETA,4=NCEP,5=NASA)
LDAS%SOIL   = 1 ! Soil parameter scheme (1=orig veg-based, 2=Reynolds 0-3.5m, 3=NLDAS
Reynolds 0-2m, 4=NLDAS Yun 0-2m)
LDAS%LAI    = 1 ! Leaf area index scheme (1=original LAI settings, 2=AVHRR dervied)
/

!===> STANDARD PARAMETERS
!===> Parameters necessary for all domains, land surface models, and forcing related to input
specifications
&ldas_run_inputs
LDAS%EXPCODE    = 999                 ! Three Digit experiment code
LDAS%NT         = 13                  ! Number of Vegetation Types
LDAS%NF         = 10                  ! Number of forcing variables
LDAS%NMIF       = 15                  ! Number of forcing variables for model init. option
LDAS%WFOR       = 0                   ! Write Forcing (0=no, 1=yes)
LDAS%WTIL       = 0                   ! Write tile space data (0=no, 1=yes
LDAS%WHDF       = 1                   ! Write HDF output files (0=no,1=yes)
LDAS%WGRB       = 0                   ! Write Grib output files (0=no,1=yes)
LDAS%WBIN       = 0                   ! Write Binary output files (0=no,1=yes)
LDAS%STARTCODE  = 3 ! **** MOS_IC,CLM_IC,CAT_IC MUST be set to the same value
as STARTCODE ****
! 1=restart file,2=realtime,3=defined,4=model init
LDAS%SSS        = 0                   ! Starting Second
LDAS%SMN        = 00                  ! Starting Minute
LDAS%SHR        = 04                  ! Starting Hour
LDAS%SDA        = 10                  ! Starting Day
LDAS%SMO        = 06                  ! Starting Month
LDAS%SYR        = 2001                ! Starting Year
LDAS%ENDCODE    = 1                   ! 0=realtime, 1=specific date
LDAS%ESS        = 0                   ! Ending Second
LDAS%EMN        = 00                  ! Ending Minute
LDAS%EHR        = 05                  ! Ending Hour
LDAS%EDA        = 10                  ! Ending Day
LDAS%EMO        = 06                  ! Ending Month
LDAS%EYR        = 2001                ! Ending Year
LDAS%TS         = 3600                ! Timestep (seconds), SHOULD NOT BE > 3600
LDAS%UDEF       = -9999.              ! Undefined value
LDAS%WRITEINTF  = 3.                  ! Forcing Output Interval (hours)
LDAS%ODIR       = "OUTPUT"            ! Output Data Base Directory
LDAS%DFILE      = "ldasdiag.dat"      ! Runtime Diagnostics File (placed in ODIR/EXP)
LDAS%FFILE      = "fsource.dat"       ! Runtime Forcing Status (placed in ODIR/EXP)
LDAS%EVTFILE    = "ETAValidTime"      ! Runtime Diag. of ETA files and Valid time
```

```
!===> Elevation Adjustment:  1=Adjust Forcing Data  0=Don't Adjust (NCEP data is already
adjusted) !
LDAS%TEMPADJ    = 1                          ! Adjust temperature data
LDAS%PRESADJ    = 1                          ! Adjust pressure data
LDAS%HUMIDADJ   = 1                          ! Adjust humidity data
LDAS%LWRADADJ   = 1                          ! Adjust long wave radiation data
LDAS%VCLASS     = 1                          ! Vegetation Classification (1=UMD)
LDAS%RPSAS      = 0                          ! Run PSAS for temperature assimilation
LDAS%RBIAS      = 0                          ! Run bias correction with assimilation
LDAS%RIBC       = 0                          ! Run incremental bias correction
LDAS%RDBC       = 0                          ! Run diurnal bias correction
LDAS%RSDBC      = 0                          ! Run semi-diurnal bias correction
LDAS%AVHRR      = "/YUKON/AVHRR_LAI"                  ! AVHRR Data directory
LDAS%ATIME      = 0.0                        ! AVHRR Time flag
/

!===> DOMAIN PARAMETER NAMELISTS
!===> Parameters that are required for running GLDAS
&gldas
LDAS%NC         = 1440                        ! Number of Columns in Grid
LDAS%NR         = 600                         ! Number of Rows in Grid
LDAS%MAXT       = 1                           ! Maximum tiles per grid
LDAS%MINA       = 0.05                        ! Min grid area for tile (%)
LDAS%VFILE      = "GVEG/UMD_60G0.25.txt"               ! Vegetation Classification Map File
LDAS%MFILE      = "GVEG/UMD60mask0.25.asc"             ! Land/Water Map File FOR
MODELLING
LDAS%FMFILE     = "GVEG/UMD60mask0.25.asc"             ! *Land/Water Map File for
FORCING DATA
LDAS%SFILE      = "BCS/sim60soil0.25.txt"              ! *Soil classification Map File
LDAS%SAFILE           = "BCS/sand60.25.bfsa"           ! Sand fraction map file
LDAS%CLFILE           = "BCS/clay60.25.bfsa"           ! Clay fraction map file
LDAS%ISCFILE          = "BCS/soicol60_2x2.5i.bin"        ! Soil color map file (for CLM)
LDAS%GPCP       = 0                           ! Global observed precipitation flag (0=Don't
Use, 1=Use)
LDAS%NRLTIME    = 0                            ! NRL precip times
LDAS%NRLDIR     = "/GLDAS1/NRL/6-hour"                ! NRL precip directory
LDAS%AGRMDIR    = "/GLDAS5/DATA/AGRMET"         ! AGRMET Forcing Base Directory
LDAS%AGRMETSW   = 0                              ! AGRMET SW Observed Radiation
(0=Don't Use, 1=Use)
LDAS%AGRMETLW   = 0                              ! AGRMET LW Observed Radiation
(0=Don't Use, 1=Use)
LDAS%KVFILE     = "GVEG/tile_info.1440x600"           ! Koster tilespace file 1/4 (not
functioning)
LDAS%KOSTER     = 0                           ! Flag to use Koster tilespace (0=no,1=yes)
LDAS%NKTYPE     = 9                           ! Number of Koster vegetation types
/
!===> LAND SURFACE MODEL PARAMETER NAMELISTS
!===> Parameters that are required for using CLM
&clm
LDAS%RCLM       = 1                          ! Run CLM (0=no,1=yes)
LDAS%WRITEINTC  = 3.                            ! CLM Output Interval (hours)
LDAS%CLM_RFILE  = "BCS/clm.rst"                    ! CLM active restart file
```

```
LDAS%CLM_VFILE  = "BCS/drv_vegm.dat"                  ! CLM Vegetation Tile Specification
File
LDAS%CLM_MVFILE = "BCS/drv_vegp.dat"                  ! CLM Vegetation Type Parameter
File
LDAS%CLM_SFILE  = "BCS/real.soilparms.txt"            ! CLM Soil Parameter File
LDAS%CLM_CFILE  = "BCS/drv_clmin.dat"                 ! CLM Constant Parameter File
LDAS%CLM_OFILE  = "BCS/clm_out.dat"                   ! CLM Output File
LDAS%CLM_PFILE  = "BCS/CLM_POUT.DAT"                  ! CLM 1D Parameter Output
File
LDAS%CLM_ISM    = 0.45                                ! CLM Initial Soil Moisture (m3/m3)
LDAS%CLM_IT     = 290.0                               ! CLM Initial Temperature (K)
LDAS%CLM_IC     = 3                                   ! CLM Initial Condition Source
LDAS%CLM_ISCV   = 0.                                  ! CLM Initial Snow Mass (kg/m2)
LDAS%CLM_SMDA   = 0                                   ! CLM SM Assimilation Option
LDAS%CLM_TDA    = 0                                   ! CLM Temperature Assimilation Option
LDAS%CLM_SDA    = 0                                   ! CLM Snow Assimilation Option
/
!===> FORCING PARAMETER NAMELISTS
!===> Parameters that are required when using GEOS forcing
&geos
LDAS%FGEOS      = 1                                   ! Use GEOS 3hr Forcing (no=0,yes=1)
LDAS%GEOSDIR    = "/GLDAS4/DATA/GEOS/BEST_LK"         ! GEOS3 Forcing Base
Directory
LDAS%ELEVFILE   = "BCS/eldif_geos60.25i.bin"          ! LDAS-GEOS Elevation Difference
File
LDAS%GEOSTIME1  = 3000.0                              ! Initial times for GEOS files
LDAS%GEOSTIME2  = 0.0
LDAS%NROLD      = 181
LDAS%NCOLD      = 360
LDAS%NMIF       = 13                                  ! Number of forcing variables for model init.
option
/
```